PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 11/00

A1

(11) International Publication Number: WO 00/33189

(43) International Publication Date: 8 June 2000 (08.06.00)

(21) International Application Number: PCT/US99/22981

(22) International Filing Date: 30 September 1999 (30.09.99)

(30) Priority Data:
09/201,111 30 November 1998 (30.11.98) US

(71) Applicant: MOTOROLA INC. [US/US]; 1303 East Algonquin Road, Schaumburg, IL 60196 (US).

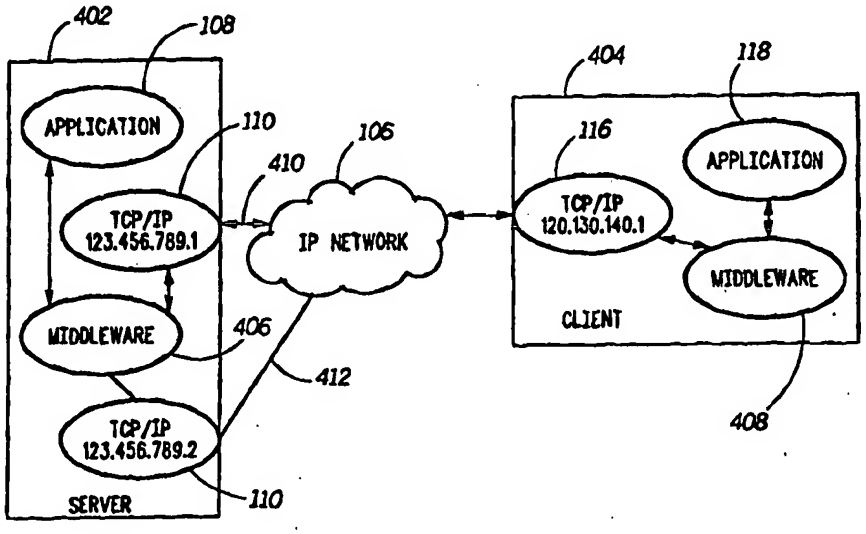
(72) Inventors: RICHARDSON, Christopher, John; 316-3480 Main Street, Vancouver, British Columbia V5V 2N2 (CA). WIEBE, Robert, John; 20-8451 Ryan Road, Richmond, British Columbia V7A 2E8 (CA).

(74) Agents: BETHARDS, Charles, W. et al.; Motorola Inc., Intellectual Property Dept., 5401 North Beach Street/MS E230, Fort Worth, TX 76137 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHOD AND APPARATUS IN A DATA COMMUNICATION SYSTEM FOR ESTABLISHING A RELIABLE INTERNET PROTOCOL SESSION



(57) Abstract

A server computer (402) and a client computer (404) establish a reliable Internet Protocol (IP) application session through a first network connection point (410) associated with a first IP address, and thereafter, determine that the first network connection point (410) is inadequate for continuing the session. In response, the server computer (402) and the client computer (404) cooperate to transfer the session to a second network connection point (412) associated with a second IP address, without interrupting the session.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD AND APPARATUS IN A DATA COMMUNICATION SYSTEM FOR ESTABLISHING A RELIABLE INTERNET PROTOCOL SESSION

Field of the Invention

This invention relates in general to Internet Protocol networking using client/server applications, and more specifically to the provision of reliable communications between a server and a client.

Background of the Invention

The Internet Protocol (IP) provides a method of establishing communication between applications on separate, networked computers. Each computer is assumed to be identifiable by one IP address; therefore, if there is a network problem that interrupts communication between the two addressable points, the applications are unable to maintain their conversation and the session terminates abnormally.

Some computer platforms, most notably those that are capable of operating in a clustered configuration, can support network connections via two or more independent IP addressable points. One such computer can simultaneously maintain IP connectivity to a network via several addresses, however the network considers each address to be a separate entity. Thus, if there is a network problem that interrupts communication on one of the addresses, affected applications still are unable to maintain their conversation and the session still terminates abnormally.

What is needed is a method and apparatus for establishing a reliable Internet Protocol (IP) session between a server application and a client application. When a network problem occurs during a session between two networked computers, both computers require a method of locating an alternate IP network connection point, re-establishing the connection, and completing the data exchange from the point of failure, without interrupting the session.

Summary of the Invention

An aspect of the present invention is a method in a data communication system for establishing a reliable Internet Protocol (IP) session between a server application and a client application. The method comprises the steps of establishing the session through a first network connection point associated with a first IP address, and thereafter, determining that the first network connection point is inadequate for continuing the session. The method further comprises, in response to the determining step, the step of transferring the session to a second network connection point associated with a second IP address, without interrupting the session.

Another aspect of the present invention is a first computer in a data communication system for establishing a reliable Internet Protocol (IP) session between a server application and a client application. The first computer comprises a network interface for establishing the session with a second computer through a network, and a processing system for controlling the session. The processing system is programmed to establish the session through a first network connection point associated with a first IP address, and thereafter, to determine whether the first network connection point is inadequate for continuing the session. The processing system is further programmed, in response to determining that the first network connection point is inadequate, to transfer the session to a second network connection point associated with a second IP address, without interrupting the session.

Another aspect of the present invention is a second computer in a data communication system for establishing a reliable Internet Protocol (IP) session between a server application and a client application. The second computer comprises a network interface for establishing the session with a first computer through a network, and a processing system for controlling the session. The processing system is programmed to establish the session through a first network connection point associated with a first IP address, and thereafter, to determine whether the first network connection point is inadequate for continuing the session. The processing system is further programmed, in response to determining that the first network connection point is inadequate, to transfer the session to a second network connection point associated with a second IP address, without interrupting the session.

Brief Description of the Drawings

FIG. 1 depicts a prior-art IP client/server application session using available protocols.

FIG. 2 depicts a prior-art IP client/server application session interrupted during failure of part of the network connection.

FIG. 3 depicts a prior-art server computer with multiple network connections communicating with a client that has one network connection.

FIG. 4 depicts an exemplary server computer with multiple network connection points communicating with a client computer in accordance with the present invention.

FIG. 5 depicts the exemplary server computer redirecting its client computer to an alternate network connection point in accordance with the present invention.

FIG. 6 depicts a simplified schematic of FIGs. 4 and 5, in which an Application Programmatic Interface (API), a Middleware Program (MP), and an Internet Protocol Network Layer (IPNL) are indicated in accordance with the present invention.

FIG. 7 depicts exemplary sets of IP addressable connection points that the server computer provides to its client computers, and exemplary connection information available to the client computer in accordance with the present invention.

FIG. 8 is an exemplary flow diagram depicting how the client computer utilizes the information it possesses about a server computer's network connection points to select a connection point and establish communication with the server computer in accordance with the present invention.

FIG. 9 is an exemplary flow diagram depicting how the server computer advertises additional network connection points to a client that has already established communication with the server computer in accordance with the present invention.

FIG. 10 is a ladder diagram depicting an example communication session between a client computer and the server computer in accordance with the present invention.

FIG. 11 depicts an exemplary list of additional server computer connection points maintained by the client computer once the server computer has made the information available.

FIGs. 12-14 are ladder diagrams depicting example communication sessions between a client and server computer in accordance with the present invention.

FIG. 15 is an electrical block diagram of an exemplary computer suitable for use either as the server computer or as the client computer, in accordance with the present invention.

Detailed Description of the Drawings

Referring now to the drawings, FIG. 1 depicts a prior-art IP client/server application session using available protocols. The session takes place through a conventional server IP network layer 110, an IP network 106, and a conventional client IP network layer 116. When a server application 108 running on the server computer 102 with a single IP network connection point 112 maintains a session with a client application 118 running on a client computer 104, the session is subject to the failure of, or congestion within, the network connection point 112, 114 at either the client or server end.

FIG. 2 depicts a prior-art IP client/server application session interrupted during failure of part of the network connection, showing the effect of such a failure when it occurs at the server network connection point 112. As a result of the indicated failure, the application session is lost and any data transfer that may have been underway at the time is terminated with an accompanying loss of data.

FIG. 3 depicts the prior-art server computer 102 with multiple network connections for communicating with the client computer 104, which has one network connection. The server computer 102 can support multiple independently-addressable IP network connection points 302. The client computer 104 must select one of the server computer's available connection points 302 to establish communication. Once a session has begun, if the selected one of the network connection points 302 fails or has congestion, the session can be interrupted and can terminate abnormally.

FIG. 4 depicts an exemplary server computer 402 with multiple network connection points 410, 412 communicating with a client computer 404 in accordance with the present invention. An Application Programmatic Interface (API) (not shown) and a middleware program 406 reside on the server computer 402, directing the application 108 to one of two available IP network connection points 410, 412 for receipt of application session requests. The middleware program 406 provides a layer of insulation between the application 108 and the multiple network connection points 410, 412, managing the provision of network

connections to the application 108. A similar API and middleware program 408 also reside on the client computer 404, managing the selection of the target network connection points 410, 412 on the server computer 402. The middleware programs 406, 408 maintain a session between the application 118 on the client computer 404 and the application 108 on the server computer 402 in case the connection point, e.g., the connection point 410, at the server end fails. In one embodiment, the communications between the server middleware program 406 and the client middleware program 408 preferably employ portions of a communication protocol known as Radio Data-Link Access Procedure (RD-LAP), available from Motorola, Inc. of Schaumburg, IL.

FIG. 5 depicts the exemplary server computer 402 redirecting its client computer 404 to an alternate network connection point 412 in accordance with the present invention. FIG. 5 is similar to FIG. 4, the essential difference being that in FIG. 5 the IP network connection point 410 has become inadequate for continuing the session, e.g., it has failed or is congested. In response, the two middleware programs 406, 408 advantageously cooperate to transfer the session through the network connection point 412, without interrupting the session. How the transfer takes place will be described further below.

FIG. 6 depicts a simplified schematic of the systems of FIGs. 4 and 5, in which an Application Programmatic Interface (API), a Middleware Program (MP), and an Internet Protocol Network Layer (IPNL) are indicated. In this drawing, the server and client applications 108, 118 are not shown, however, any application running on the client computer 404 with a session to another application on the server computer 402 will utilize the API 604 to the middleware program 408, which then manages the IPNL 116. Similarly, any application running on the server computer 402 with a session to another application on the client computer 404 will utilize the API 602 to the middleware program 406, which then manages the IPNL 110.

FIG. 7 depicts exemplary sets of IP addressable connection points that the server computer 402 provides to its client computers 404, and exemplary connection information available to the client computers 404 in accordance with the present invention. The server computer IP addresses are split into two groups: the Static Computer Addresses (SCAs) 706 and the Dynamic Computer Addresses (DCAs) 708. The middleware program 408 on the client computer 404 has access to local configuration information (e.g., a host name) such that it can determine an SCA table 702 comprising SCAs belonging to the server computer 402 using standard techniques (e.g. a domain name server, a hosts file, etc.). In this example,

a domain name server 704 maps the SCA host names from the SCA table 702 on the client computer 404 to an IP address belonging to the SCAs 706 on the server computer 402. The domain name server 704 also makes the set of DCAs 708 available to the client computer 404. The server computer 402 will provide the list of available DCAs 708 to the client computer 404 once a connection has been established using one of the SCAs 706.

When an application on the client computer 404 wishes to establish communication with an application on the server computer 402, it issues an API "Connect" request. The API 604 passes the request to the middleware program 408, which then follows an algorithm to determine which of the available SCA IP addressable connection points provided by the server computer 402 will be used.

FIG. 8 is an exemplary flow diagram depicting how the client computer 404 utilizes the information it possesses about a server computer's network connection points to select a connection point and establish communication with the server computer 402 in accordance with the present invention. First, the client computer 404 examines the SCA table 702 to select 802 the next available SCA host identifier. The client computer 404 then uses well known techniques to resolve 804 the SCA host identifier into an IP address. The client computer next uses additional well-known techniques to attempt 806 to establish a connection with the server computer's IP addressable connection point identified by the IP address just resolved. At step 808 the client computer 404 determines whether the connection attempt was successful. If successful, the client computer 404 proceeds to step 812; if a time-out or error occurred instead, the client computer 404 proceeds to step 810.

At step 812, the connection has been established, and the client computer sends 812 a success indication to the application on the client computer 404, and then sends 814 a connection acknowledgment to the server computer 402. At step 810, the client computer checks whether there are additional, unused SCAs in the SCA table 702. If so the client computer 404 returns to step 802; otherwise the client computer 404 proceeds to step 816, where it sends a failure indication to the client application.

Once an application on the client computer 404 has established communication with an application on the server computer 402, the server computer's middleware program 406 can "advertise" a set of DCAs to the client computer's middleware program 408. The server computer's MP configuration may change at any time, so it also may advertise an updated DCA list to the client computer's middleware program 408 at any time. It will be appreciated that the

SCA and DCA lists preferably are made fully available to the client middleware program 408 each time a new or replacement connection is made. More specifically, any state information associated with an SCA or a DCA indicating that a connection attempt to that address has failed during a previous execution of the flow chart of FIG. 8 should be cleared before the client computer 404 attempts to establish or re-establish a connection with the server computer 402.

FIG. 9 is an exemplary flow diagram depicting how the server computer 402 advertises additional network connection points to a client computer 404 that has already established communication with the server computer 402, in accordance with the present invention. First, the server computer 402 uses well-known techniques to accept 902 a connection request from the client computer 404 and to establish communication therewith. The server computer 402 then sends 904 a list of DCAs 708 to the client computer 404. The server computer 402 periodically determines 906 whether the DCA list on the server computer 402 has been updated. If it has been updated, the server computer determines 908 whether the client computer 404 is still connected to the server computer 402. If the connection is still available, the flow returns to step 904 to send the DCA list; otherwise the process ends. If, on the other hand, at step 906 the DCA list has not been updated, the server computer 402 repeats step 906 (after a predetermined waiting period).

FIG. 10 is a ladder diagram 1000 depicting an example communication session between a client computer and the server computer in accordance with the present invention. In the diagram, time progresses forward down the vertical axes, as depicted by the time line 1006. A server middleware line 1002 depicts events generated and received by the server middleware program 406, while a client middleware line 1004 depicts events generated and received by the client middleware program 408. First, the client middleware program 408 locates an SCA IP network connection point, SCA1, from its SCA table 702 and attempts to connect to it. In this case, the attempt fails, when the client middleware program 408 determines that a time-out has occurred on the connection attempt. Next, the client middleware program 408 locates the next SCA IP connection point, SCA2, from its SCA table 702 and attempts to connect to it. In this case, the connection attempt succeeds. In response, the server middleware program 406 issues a successful connection message (OK) to the client middleware program 408. The client middleware program 408 then issues an acknowledgment of receipt of the successful connection message to the server middleware program 406. The server middleware program 406 then sends its list of DCA IP connection points to the

client middleware program 408. At some point in time during the connection, the server middleware program 406 determines that its DCA list has been updated. In response, the server middleware program 406 sends its list of DCA IP connection points to the client middleware program 408.

FIG. 11 depicts an example DCA table 1102 comprising additional server computer connection points maintained by the client computer 404 once the server computer 402 has made the information available to the client computer 404. The client computer 404 has successfully established a connection to the server computer 402 via the IP addressable connection point referenced by the entry named "SCA2" in the SCA table 702 maintained by the client middleware program 408. After the connection was made, the server computer 402 sent a new list of IP addressable connection points to the client computer 404 which then stored the information in its DCA table 1102.

The algorithms detailed in FIGs. 8 and 9 can be employed to provide the client middleware program 408 with the ability to "roam" between available IP addressable connection points on the server computer 402. When failure or congestion within an established connection is discovered by the client middleware program 408, it can then attempt to establish a connection to one of the alternate IP addressable points referenced by entries in either the SCA or DCA tables 702, 1102. The server and client applications are unaware of the changed connection point.

FIGs. 12-14 are ladder diagrams 1200, 1300, 1400 depicting example communication sessions between the client and server computers in accordance with the present invention. The diagram 1200 demonstrates a typical exchange between the client and server computers 404, 402 employing the middleware embodiment of this disclosure. In this example, a connection fails while the client and server applications are performing a data transfer. The recovery is handled by the middleware programs 406, 408 in such a way that the applications are never aware that a failure has occurred. In this diagram, time progresses forward down the vertical axes, as indicated by the time line 1208. A first server middleware line 1202 depicts events generated and received by the server middleware program 406 through the IP connection point SCA2, while a second server middleware line 1206 depicts events generated and received by the server middleware program 406 through the IP connection point.

Using the algorithm described in FIG. 8, the client middleware program 408 locates the IP connection point, SCA2, from its SCA table and establishes a connection to it; the server middleware program 406 issues the successful

connection message (OK) to the client middleware program 408; and the client middleware program 408 sends an acknowledgment of receipt of the successful connection message to the server middleware program 406. The server middleware program 406 then sends its list of DCA IP connection points to the client middleware program 408. The client application 118 issues an API "Send" request to the client middleware program 408 to transmit its first set of data to the server application 108. The server middleware program 406 then acknowledges receipt of the first set of data and forwards the data to the server application 108. The client application 118 issues an API "Send" request to the client middleware program 408 to transmit its second set of data to the server application 108. The server middleware program 406 acknowledges receipt of the second set of data and forwards the data to the server application 108. The connection between the client computer and the server computer at the IP addressable connection point identified by SCA2 then fails. The client application 118 issues an API "Send" request to the client middleware program 408 to transmit its third set of data to the application on the server computer. The client middleware program 408 makes several attempts to transmit the data to the server computer 402 but is unsuccessful due to the connection failure. The middleware programs 406, 408 preferably provide a configurable setting for the number of transmissions that will be attempted before a connection is deemed to have failed. They also preferably provide a configurable setting for the delay between transmission attempts for the same set of data. Having determined that the connection to SCA2 is no longer available, the client middleware program 408 selects an alternate connection point on the server computer 402 from its SCA and DCA tables 702, 1102. Using the algorithm described in FIG. 8, the client middleware program 408 locates the IP connection point, DCA1, from its DCA table 1102 and establishes a connection to it; the server middleware program 406 issues the successful connection message (OK); and the client middleware program 408 acknowledges the connection. The client middleware program 408 then completes the API "Send" request for the third set of data by successfully transmitting the data to the new connection point, DCA1. The server middleware program 406 acknowledges receipt of the third set of data and forwards the data to the application 108 on the server computer 402.

The ladder diagram 1300 demonstrates an exchange between client and server computers employing the middleware embodiment of this disclosure. In this example, the client computer 404, while in the middle of data transfer, determines that the original connection point is congested and roams to an alternate connection point. The change is handled by the middleware programs 406, 408 in such a way

that the applications are never aware that the connection point has switched. Using the algorithm described in FIG. 8, the client middleware program 408 locates the IP connection point, SCA2, from its SCA table and establishes a connection to it; the server middleware program 406 issues the successful connection message (OK) to the client middleware program 408; and the client middleware program 408 sends an acknowledgment of receipt of the successful connection message to the server middleware program 406. The server middleware program 406 then sends its list of DCA IP connection points to the client middleware program 408. The client application 118 then issues an API "Send" request to the client middleware program 408 to transmit its first set of data to the server application 108. The server middleware program 406 acknowledges receipt of the first set of data and forwards the data to the server application 108. The client application 118 issues an API "Send" request to the client middleware program 408 to transmit its second set of data to the server application 108. The server middleware program 406 acknowledges receipt of the second set of data and forwards the data to the server application 108. The client middleware program 408 detects congestion within the SCA2 connection point and initiates the connection algorithm. The client middleware program 408 sends a "Roam" notification to the server computer 402 to indicate that it will be changing connection points. The client middleware program 408 uses the algorithm described in FIG. 8 to locate a new IP connection point from its SCA and DCA tables 702, 1102. In this example, it successfully locates the connection point DCA1, and establishes a connection to it. The server middleware program 406 issues a successful connection message (OK) and the client middleware program 408 issues an acknowledgment. The client middleware program 408 completes the API "Send" request for the third set of data by successfully transmitting the data to the new connection point, DCA1. The server middleware program 406 acknowledges receipt of the third set of data and forwards the data to the server application 108.

The algorithms detailed in FIGs. 8 and 9 can be employed by the server computer 402 to command the client computer 404 to "roam" to an alternate IP addressable connection point. The server computer 402 can specify a DCA or SCA connection point in the connection change command.

The ladder diagram 1400 demonstrates an exchange between client and server computers 404, 402 employing the middleware embodiment of this disclosure. In this example, the server computer 402, while in the middle of data transfer, determines that the original connection point is congested and directs the client computer 404 to roam to an alternate connection point. The change is handled by

the middleware programs 406, 408 in such a way that the applications 108, 118 are never aware that the connection point has switched.

Using the algorithm described in FIG. 8, the client middleware program 408 locates the IP connection point, SCA2, from its SCA table and establishes a connection to it; the server middleware program 406 issues the successful connection message (OK) to the client middleware program 408; and the client middleware program 408 sends an acknowledgment of receipt of the successful connection message to the server middleware program 406. The server middleware program 406 then sends its list of DCA IP connection points to the client middleware program 408. The client application 118 issues an API "Send" request to the client middleware program 408 to transmit its first set of data to the server application 108. The server middleware program 406 acknowledges receipt of the first set of data and forwards the data to the server application 108. The client application 118 issues an API "Send" request to the client middleware program 408 to transmit its second set of data to the server application 108. The server middleware program 406 acknowledges receipt of the second set of data and forwards the data to the server application 108. The server middleware program 406 then detects congestion within the SCA2 connection point and issues a "Go To" command to the client middleware program 408. The server middleware program 406 can provide a new connection point identifier or can allow the client middleware program 408 to select the next available connection point from its SCA and DCA tables 702, 1102. In this example, the server computer 402 provides a specific connection point identifier, DCA1. The client middleware program 408 uses the algorithm described in FIG. 8, starting with the connection point identifier provided by the server middleware program 406, to locate a new IP connection point. In this example, it successfully locates the connection point DCA1, and establishes a connection to it. Had the connection attempt failed, the client computer would have sought the next available connection point from its SCA and DCA tables 702, 1102. The server middleware program 406 issues a successful connection message (OK) and the client middleware program 408 issues an acknowledgment. The client middleware program 408 completes the API "Send" request for the third set of data by successfully transmitting the data to the new connection point, DCA1. The server middleware program 406 acknowledges receipt of the third set of data and forwards the data to the server application 108.

It will be appreciated that applications utilizing the middleware programs 406, 408 preferably conform to an API (not a specific API since several APIs can be layered on the same MP, and it is possible that the client and server could use

different APIs). It will be further appreciated that the invention disclosed here can be extended from a single computer that supports multiple, independent IP addressable network connection points to a set of independent computers each with their own IP addressable network connection point(s). The middleware embodiment of this invention can be utilized to collect a group of such IP addressable computers into a set of related IP addressable network connection points, such that application sessions could be transferred between independent computers rather than between network connection points on the same computer. This extension would require an additional set of middleware that would ensure that the context of an application session could be transferred from one computer to another when the connection point is changed.

FIG. 15 is an electrical block diagram of an exemplary computer 402, 404 suitable for use either as the server computer 402 or as the client computer 404, in accordance with the present invention. The computer 402, 404 comprises a conventional network interface 208 for establishing an application session with another computer through a network. The computer 402, 404 further comprises a processing system 206 coupled to the network interface for controlling the session. A user interface 214 is coupled to the processing system 206 for interfacing with a user. The user interface 214 includes a display 216, such as a conventional video display, and a conventional keyboard 220. The processing system 206 comprises a conventional processor 210 coupled to a conventional memory 212, which preferably includes a mix of ROM, RAM, and magnetic disk memory elements. The memory 212 includes software programs and other data for programming the processing system 206 in accordance with the present invention. The memory 212 comprises at least one application 108, 118 for establishing the application session with a similar application in another computer through a network. The memory 212 further comprises the Application Programmatic Interface (API) 602,604 for programming the processing system 206 to interface with the application 108, 118 in a predetermined manner. The memory 212 also includes the middleware program 406, 408 for programming the processing system 206 to establish and maintain a reliable Internet Protocol (IP) session between the server application 108 and the client application 118, in accordance with the present invention. In addition, the memory 212 includes the Internet Protocol Network Layer (IPNL) 110, 116 for interfacing with the network. The memory 212 further comprises an SCA store 230 and a DCA store 232 for storing Static Computer Addresses and Dynamic Computer Addresses, respectively.

The preceding examples have described a server computer having redundant links to the network, communicating with a client computer having a non-redundant link to the network. It will be appreciated that, alternatively, the roles can be reversed. That is, the client computer can have redundant links and can advertise DCAs, while the server computer has a single non-redundant link and stores SCAs and DCAs. It will be further appreciated that, as another alternative for even better link reliability, both the client computer and the server computer can have redundant links to the network and each can incorporate the link reliability features of both the server computer and the client computer, as described herein above, in accordance with the present invention.

Thus, it should be clear from the preceding disclosure that the present invention provides a method and apparatus for establishing a reliable Internet Protocol (IP) session between a server application and a client application. When a network problem occurs during a session between two networked computers, the present invention advantageously provides a technique for locating an alternate IP network connection point, re-establishing the connection, and completing the data exchange from the point of failure.

Many modifications and variations of the present invention are possible in light of the above teachings. For example, alternatively, the link redundancy can be performed with SCAs exclusively, without the use of DCAs and advertising. Thus, it is to be understood that, within the scope of the appended claims, the invention can be practiced other than as specifically described herein above.

What is claimed is:

CLAIMS

1. A method in a data communication system for establishing a reliable Internet Protocol (IP) session between a server application and a client application, the method comprising the steps of:

establishing the session through a first network connection point associated with a first IP address;

thereafter, determining that the first network connection point is inadequate for continuing the session; and

in response to the determining step, transferring the session to a second network connection point associated with a second IP address, without interrupting the session.

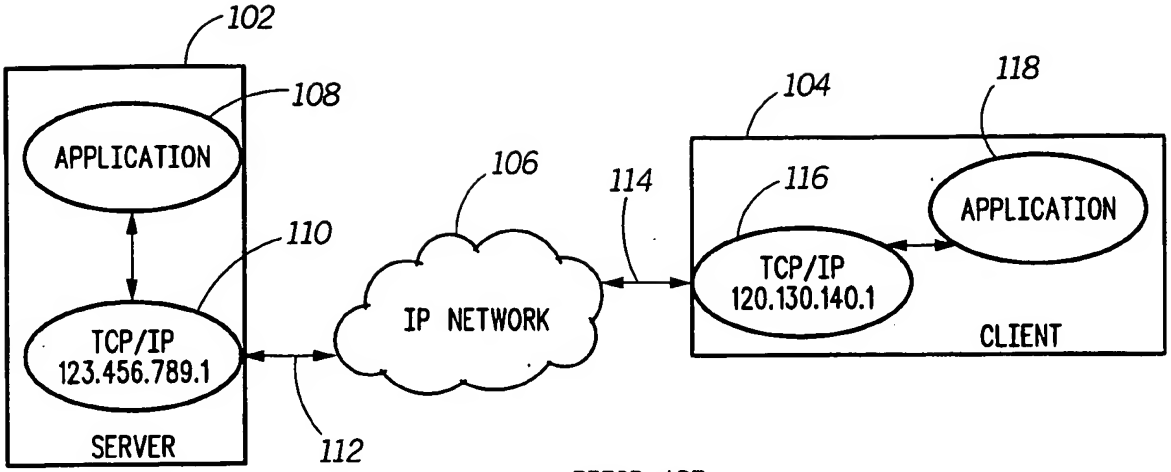
2. The method of claim 1, further comprising, after the establishing step, the step of

advertising, by a computer associated with the first and second IP addresses, at least one available dynamic computer address (DCA) that can be used as the second IP address.

3. The method of claim 1,

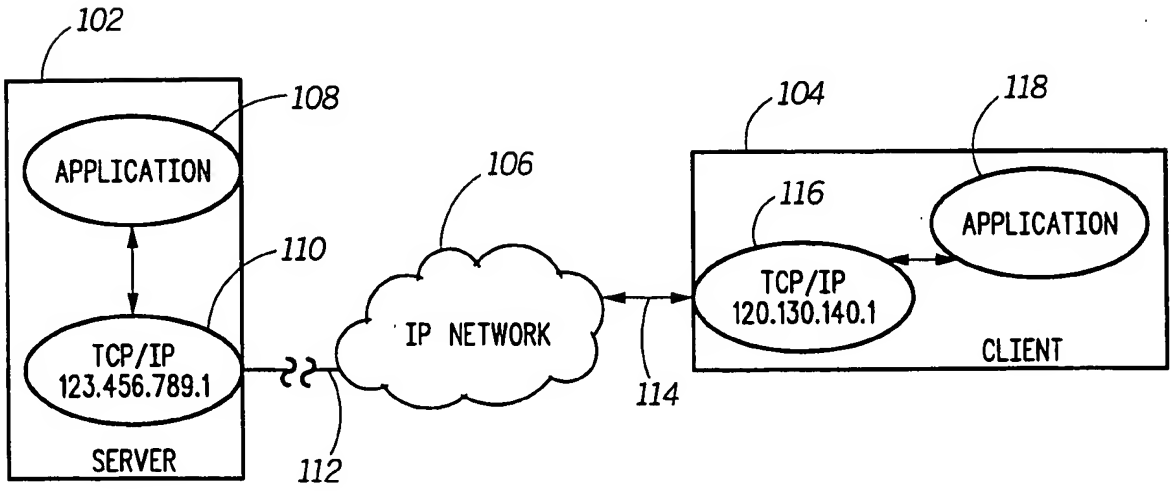
wherein the determining step is performed by a middleware program, and

wherein the transferring step comprises the step of establishing a connection with the second network connection point by the middleware program.



PRIOR ART

FIG. 1



PRIOR ART

FIG. 2

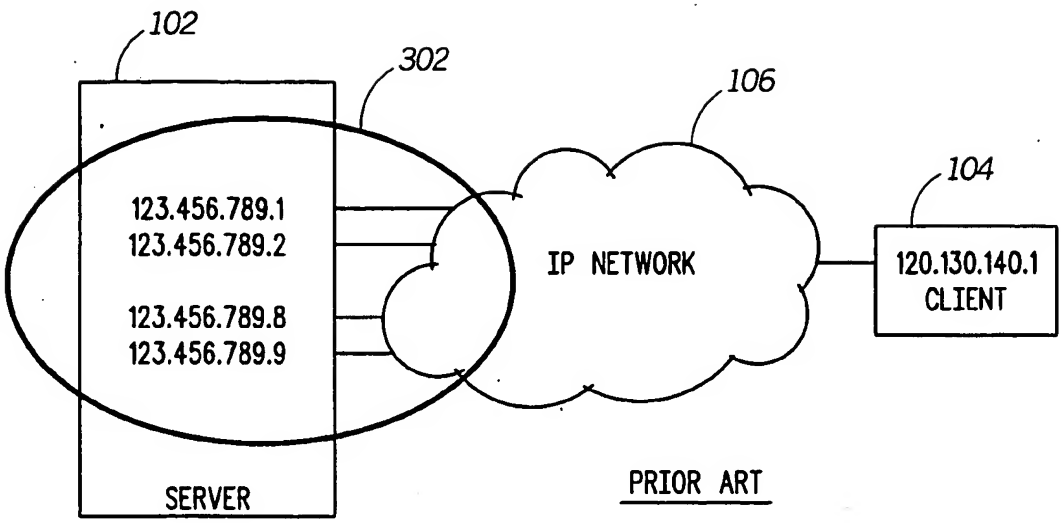

102
302
106
104
123.456.789.1
123.456.789.2
123.456.789.8
123.456.789.9
IP NETWORK
120.130.140.1
CLIENT
SERVER
PRIOR ART

FIG. 3

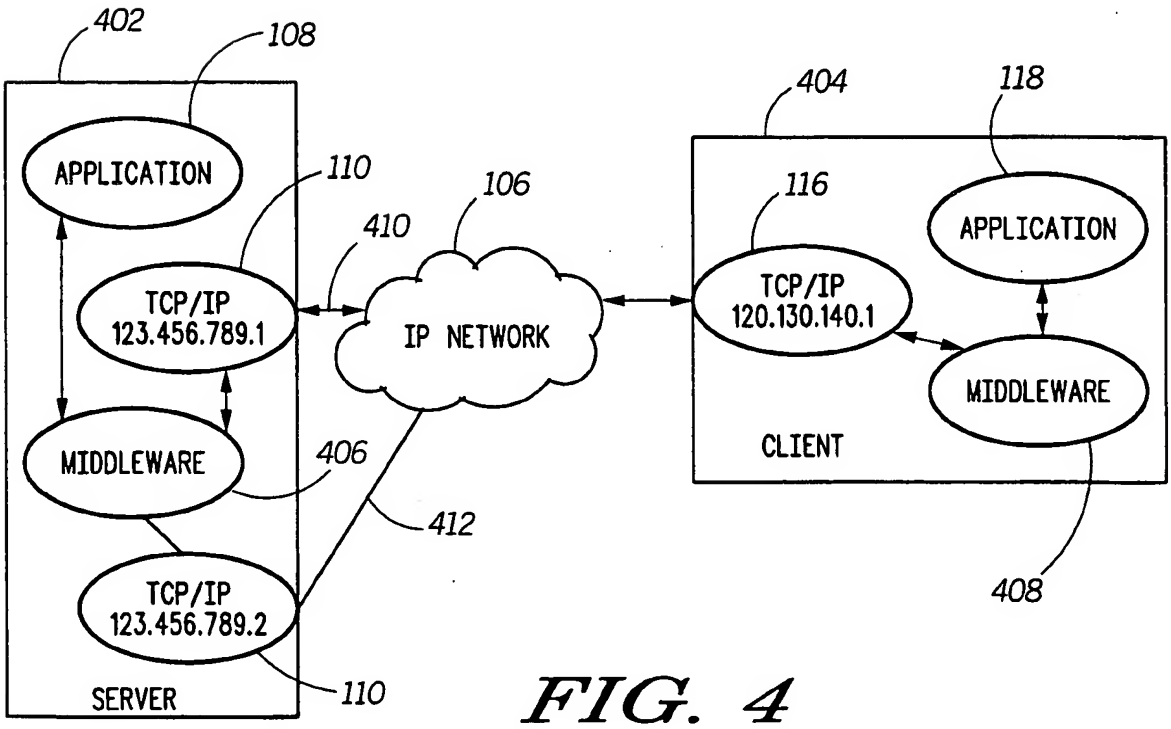

402
108
404
118
APPLICATION
110
106
116
APPLICATION
410
TCP/IP
123.456.789.1
IP NETWORK
TCP/IP
120.130.140.1
MIDDLEWARE
CLIENT
MIDDLEWARE
406
412
TCP/IP
123.456.789.2
408
SERVER
110

FIG. 4

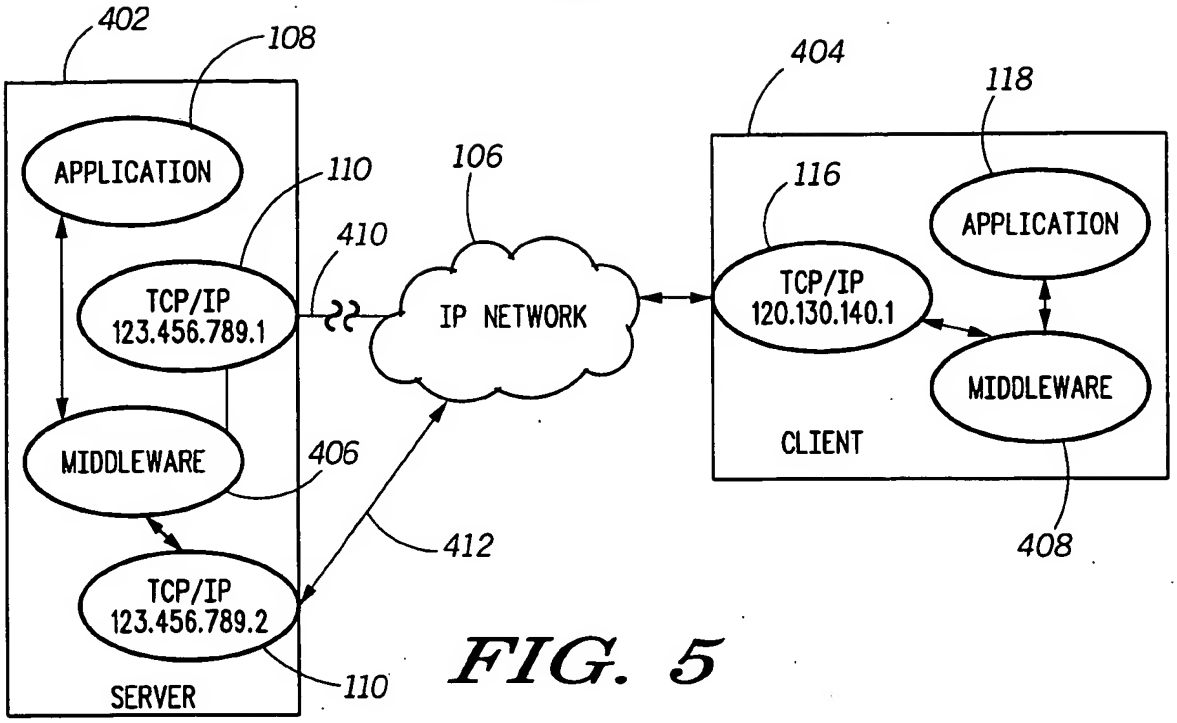

402
108
APPLICATION
110
TCP/IP
123.456.789.1
410
106
IP NETWORK
404
118
116
APPLICATION
TCP/IP
120.130.140.1
MIDDLEWARE
CLIENT
MIDDLEWARE
406
412
TCP/IP
123.456.789.2
110
SERVER
408

FIG. 5

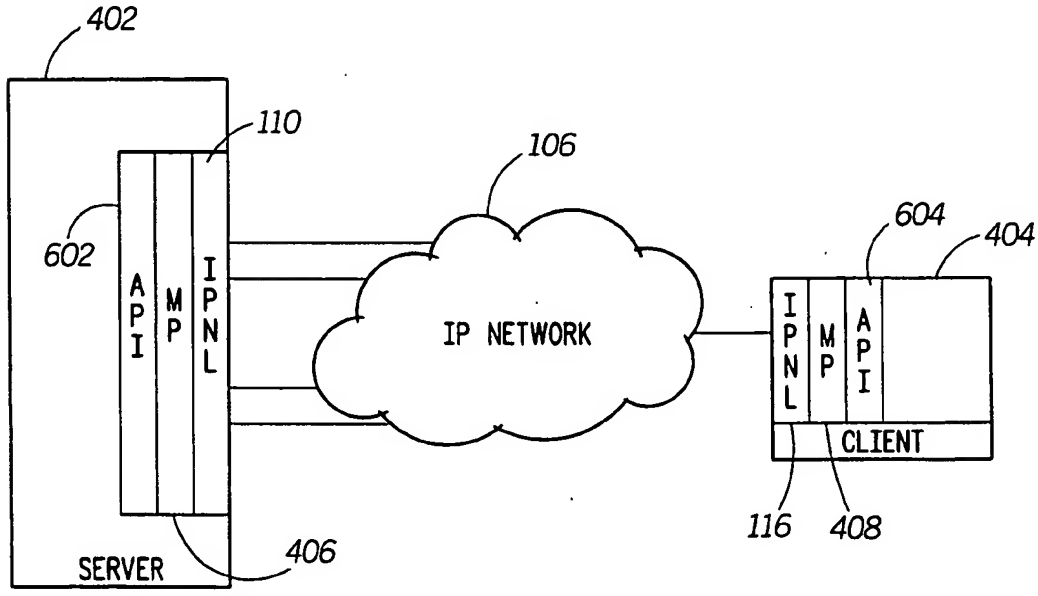

402
110
106
602
604
404
API
MP
IPNL
IP NETWORK
IPNL
MP
API
CLIENT
SERVER
406
116
408

FIG. 6

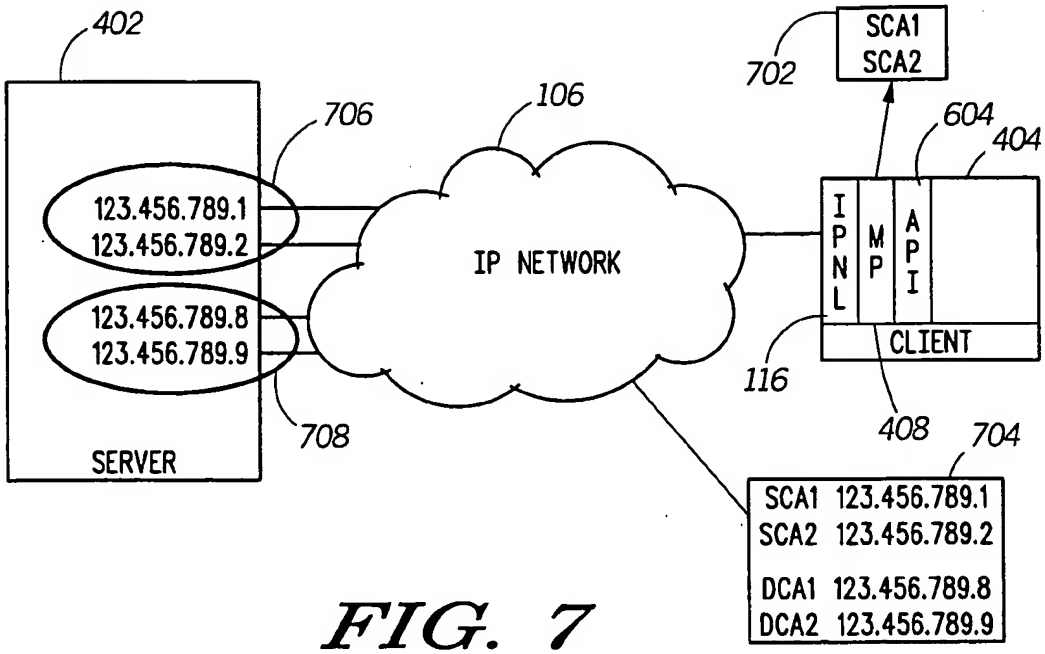

402
706
106
702
SCA1
SCA2
604
404
123.456.789.1
123.456.789.2
123.456.789.8
123.456.789.9
IP NETWORK
IPNL
MP
API
CLIENT
116
408
704
708
SERVER
SCA1 123.456.789.1
SCA2 123.456.789.2
DCA1 123.456.789.8
DCA2 123.456.789.9

FIG. 7

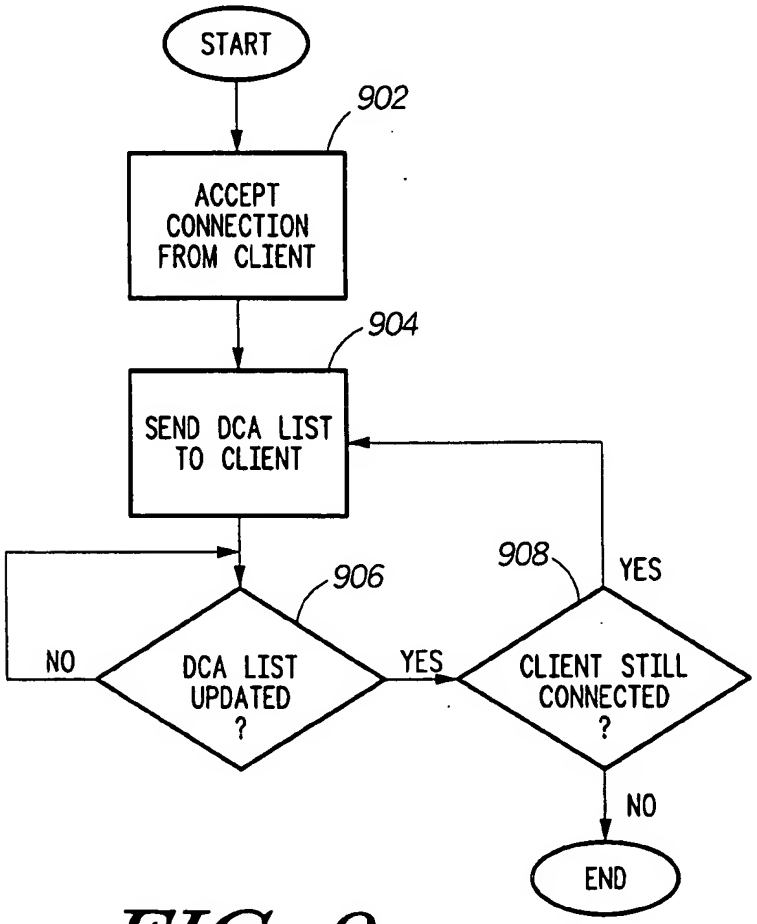

START
902
ACCEPT CONNECTION FROM CLIENT
904
SEND DCA LIST TO CLIENT
906
908
YES
NO
DCA LIST UPDATED ?
YES
CLIENT STILL CONNECTED ?
NO
END

FIG. 9

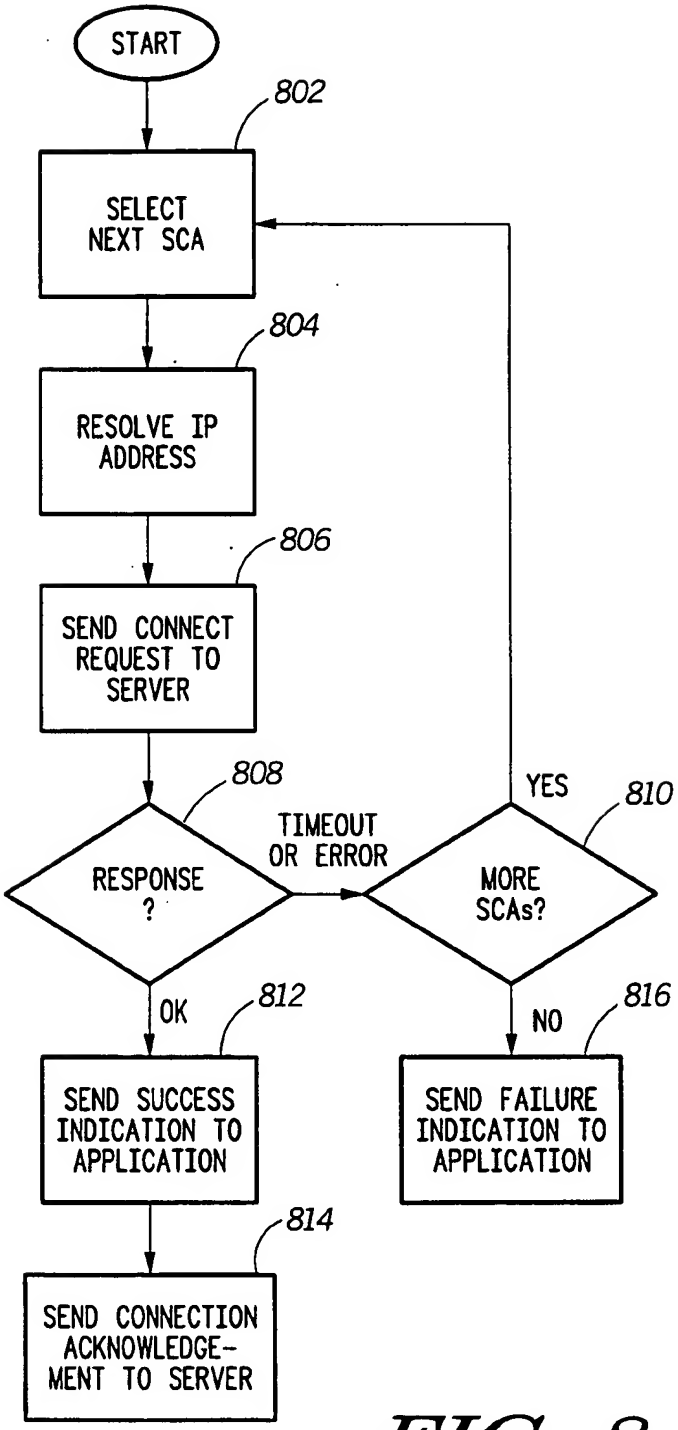

START
802
SELECT NEXT SCA
804
RESOLVE IP ADDRESS
806
SEND CONNECT REQUEST TO SERVER
808
RESPONSE ?
TIMEOUT OR ERROR
YES
810
MORE SCAs?
OK
812
NO
816
SEND SUCCESS INDICATION TO APPLICATION
SEND FAILURE INDICATION TO APPLICATION
814
SEND CONNECTION ACKNOWLEDGE-MENT TO SERVER

FIG. 8

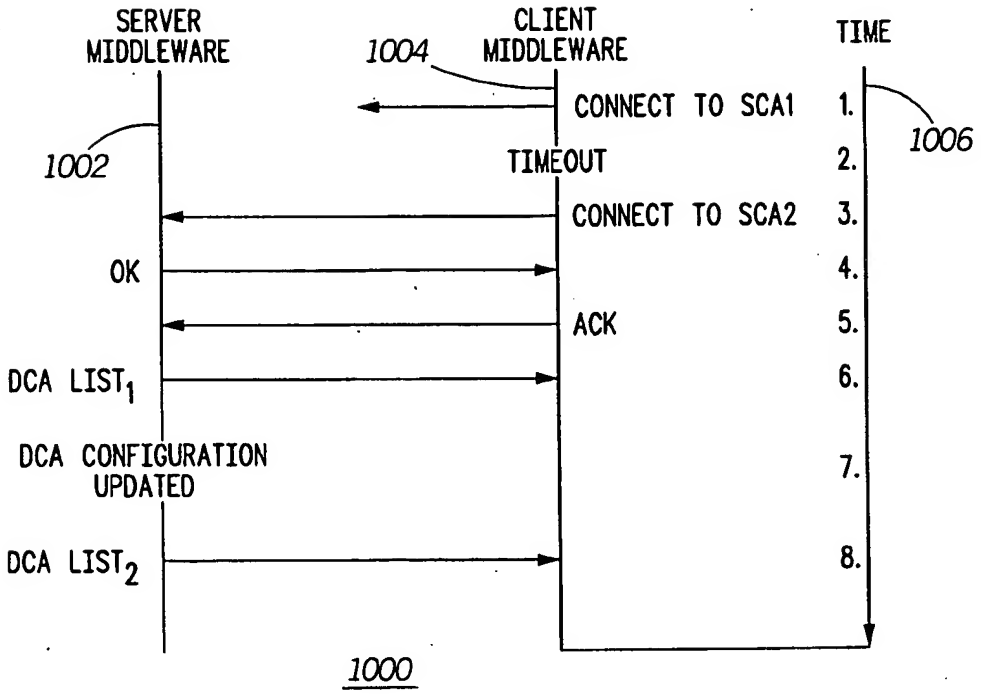

SERVER MIDDLEWARE
CLIENT MIDDLEWARE
TIME
1004
1002
1006
CONNECT TO SCA1
TIMEOUT
CONNECT TO SCA2
OK
ACK
DCA LIST1
DCA CONFIGURATION UPDATED
DCA LIST2
1.
2.
3.
4.
5.
6.
7.
8.
1000

FIG. 10

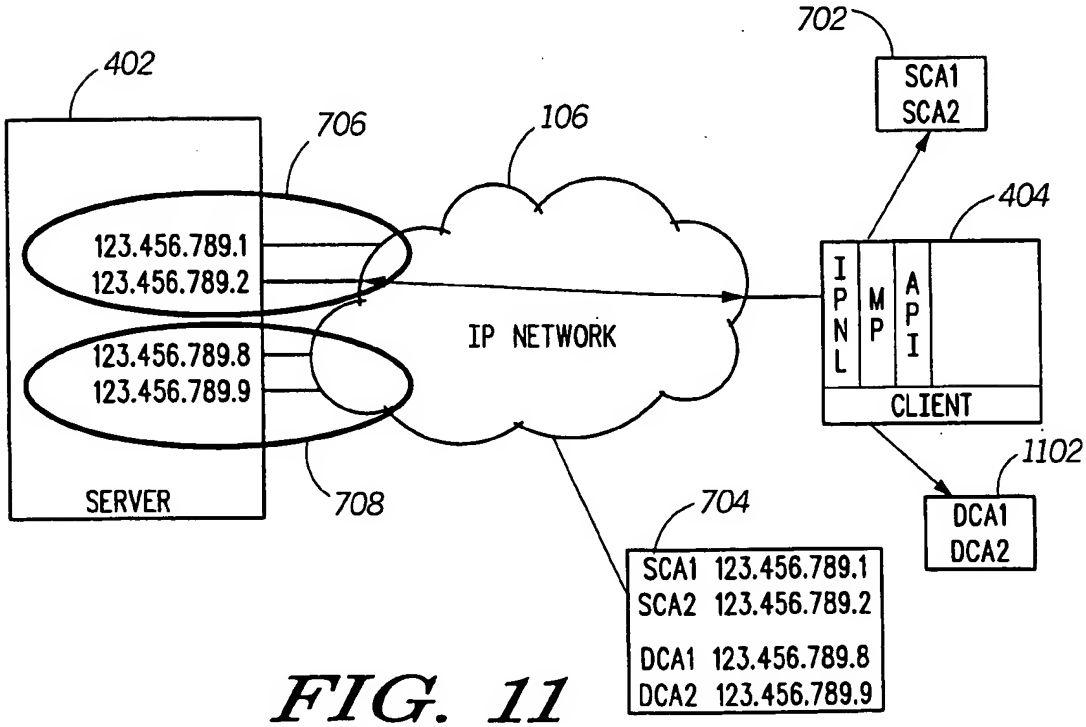
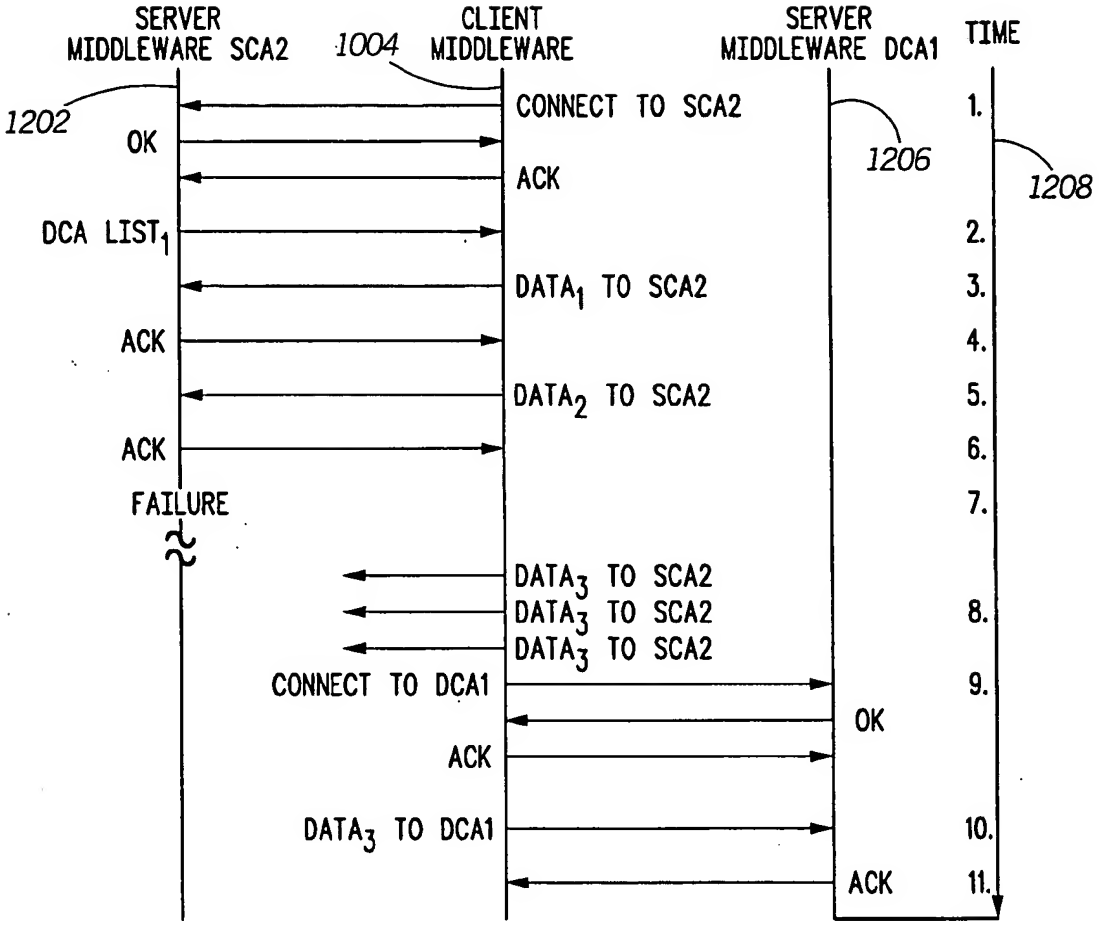
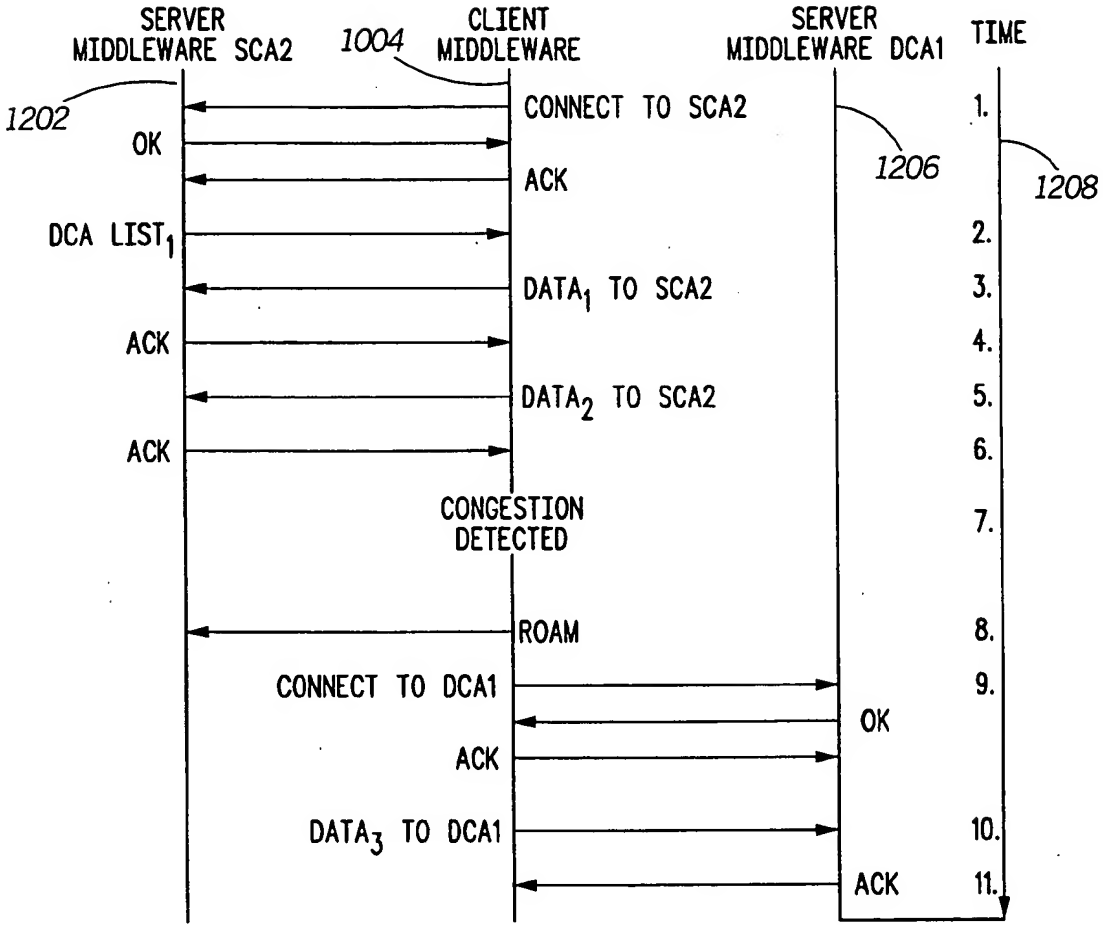

702
SCA1
SCA2
402
706
106
404
123.456.789.1
123.456.789.2
123.456.789.8
123.456.789.9
IP NETWORK
IPNL
MP
API
CLIENT
SERVER
708
704
1102
DCA1
DCA2
SCA1 123.456.789.1
SCA2 123.456.789.2
DCA1 123.456.789.8
DCA2 123.456.789.9

FIG. 11


SERVER MIDDLEWARE SCA2
1004
CLIENT MIDDLEWARE
SERVER MIDDLEWARE DCA1
TIME
1202
CONNECT TO SCA2
1.
OK
1206
ACK
1208
DCA LIST1
2.
DATA1 TO SCA2
3.
ACK
4.
DATA2 TO SCA2
5.
ACK
6.
FAILURE
7.
DATA3 TO SCA2
DATA3 TO SCA2
8.
DATA3 TO SCA2
CONNECT TO DCA1
9.
OK
ACK
DATA3 TO DCA1
10.
ACK
11.

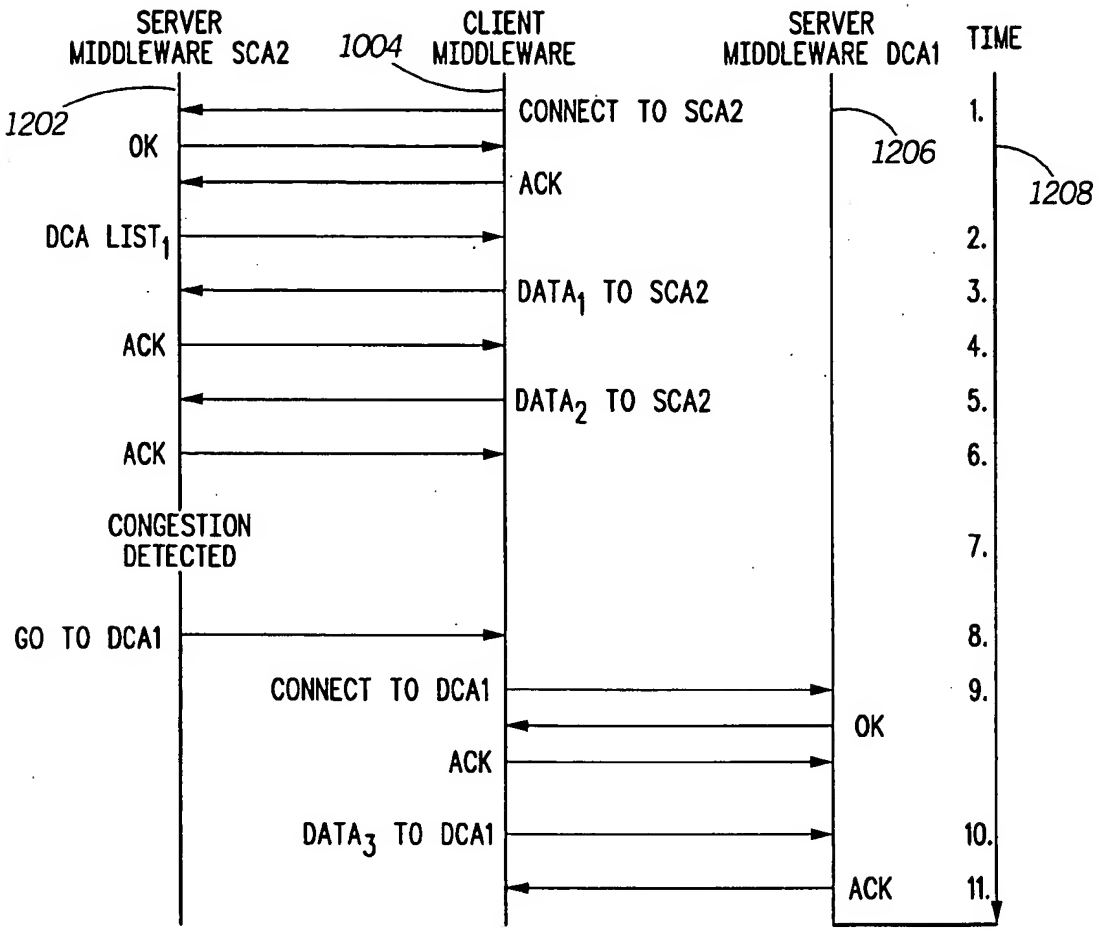
1200

FIG. 12


SERVER MIDDLEWARE SCA2
1004
CLIENT MIDDLEWARE
SERVER MIDDLEWARE DCA1
TIME
1202
CONNECT TO SCA2
1.
OK
ACK
1206
1208
DCA LIST1
2.
DATA1 TO SCA2
3.
ACK
4.
DATA2 TO SCA2
5.
ACK
6.
CONGESTION DETECTED
7.
ROAM
8.
CONNECT TO DCA1
9.
OK
ACK
DATA3 TO DCA1
10.
ACK
11.

1300

FIG. 13


SERVER MIDDLEWARE SCA2
1004
CLIENT MIDDLEWARE
SERVER MIDDLEWARE DCA1
TIME
1202
1206
1208
CONNECT TO SCA2
1.
OK
ACK
DCA LIST1
2.
DATA1 TO SCA2
3.
ACK
4.
DATA2 TO SCA2
5.
ACK
6.
CONGESTION DETECTED
7.
GO TO DCA1
8.
CONNECT TO DCA1
9.
OK
ACK
DATA3 TO DCA1
10.
ACK
11.

1400

FIG. 14

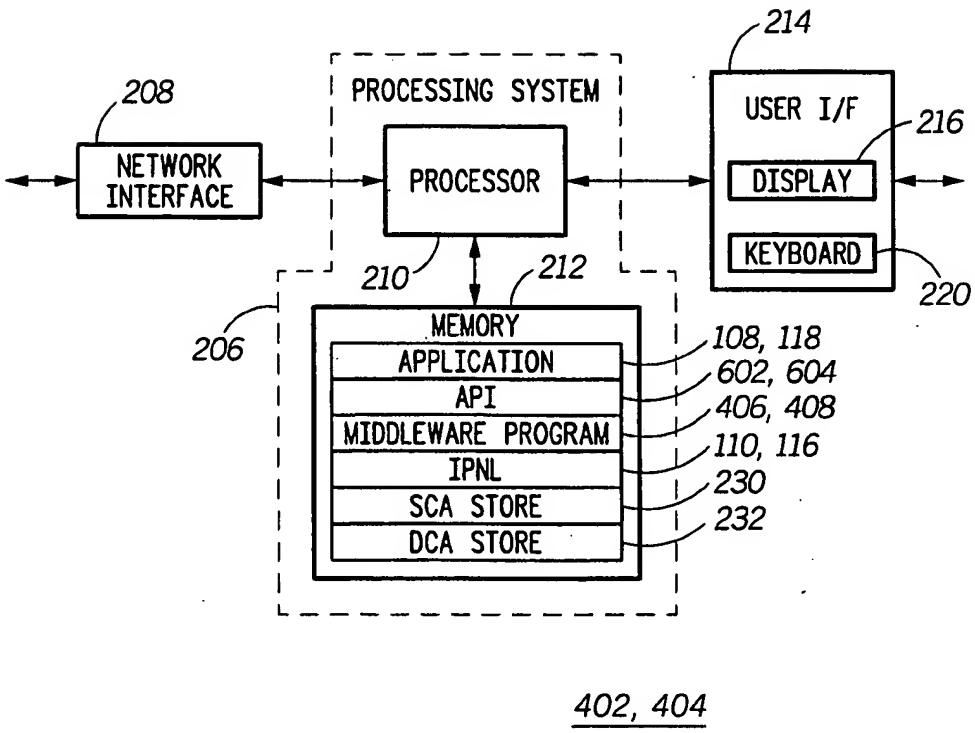

214
208
PROCESSING SYSTEM
USER I/F
216
NETWORK INTERFACE
PROCESSOR
DISPLAY
KEYBOARD
220
210
212
206
MEMORY
APPLICATION
API
MIDDLEWARE PROGRAM
IPNL
SCA STORE
DCA STORE
108, 118
602, 604
406, 408
110, 116
230
232
402, 404

FIG. 15

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/22981

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 11/00
US CL : 395/575

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/575, 800; 709/245; 370/216; 364/489; 714/4

Documer .on searched other than minimum documentation to the extent that such documents ar .cluded in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,426,773 A (CHABANET et al) 20 June 1995, col. 3, line 27-col. 9, line 38.	1-21
Y	US 5,661,719 A (TOWNSEND et al) 26 August 1997, col. 2, lines 1-22.	1-21
Y,E	US 6,009,474 A (LU et al) 28 December 1999, col. 1, lines 1-68.	1-21
Y	US 5,812,413 A (MATSUSHITA) 22 September 1998, col. 3-col. 5, lines 1-68.	1-21
Y, P	US 5,948,108 A (LU et al) 07 September 1999, col. 1, lines 38-68-col. 2, line 47.	1-21
Y	US 5,652,908 A (DOUGLAS et al) 29 July 1997, col. 1, line16-col. 3, line3.	1-21

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot b. considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
04 JANUARY 2000	07 FEB 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer HIEU C. LE Telephone No. (703) 306-3101